SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

OBJECT PERSISTENCE TO RELATIONAL DATABASE WITHIN RUN-TIME ENVIRONMENT SUPPORTING ATTRIBUTES AND REFLECTION

Background of Invention

[0001] This invention relates generally to object-oriented software objects, and more particularly to persisting such objects within relational database management systems (RDBMS).

[0002] Object-oriented technology in conjunction with software can be generally defined as the use of objects as the building blocks for applications. Objects are independent program modules written in object-oriented programming languages. Just as hardware components are routinely designed as modules to plug into and work with each other, objects are software components designed to work together at runtime without any prior linking or precompilation as a group.

[0003] In object-oriented application development, it is often desirable to store an object's current state into a permanent storage medium. Persistence may be needed to free up software and hardware resources used by such an object, when the object is no longer currently being used. When the object is later needed, it may then be retrieved from the permanent storage medium. The storage of objects into a permanent storage medium is generally referred to as persistence.

[0004] One type of permanent storage medium that is well known, and therefore has been looked to for object persistence, is a relational database management system

(RDBMS). A relational database links files together as required. Relationships between files are created by comparing data, such as account numbers and names. A relational system has the flexibility to take any two or more files and generate a new file from the records that meet the matching criteria.

[0005] Relational databases are an example of non-object-oriented data stores. As such, they organize and store data differently than do object-oriented paradigms, such as software objects. Relational databases are thus inherently incapable of communicating with software objects in object-oriented terms, and are not immediately amenable as a permanent storage medium for the persistence of software objects. This is unfortunate, since relational databases represent established and known technology.

[0006] Current approaches to allow software objects to persist to relational databases and other types of non-object-oriented data stores have focused on external bridging mechanisms that enable software objects to communicate with such data stores. However, even when these mechanisms are available, they can require significant programming effort on the part of developers to make their software objects persistent using a non-object-oriented data store. The additional complexity of the mechanisms may become burdensome to the developers.

[0007] For example, a software object may first have to establish a communication channel with a non-object-oriented data store through an external bridging mechanism. The object may then have to translate object states and relationships into commands and/or languages, such as the Structure Query Language (SQL), understandable by the non-object-oriented data store. Finally, the object may have to issue such commands and/or languages to the non-object-oriented database to achieve the desired persistence functionality.

[0008] Therefore, the presence of an external bridging mechanism is no panacea to the average developer of software objects using object-oriented technology. The developer must still have an understanding of how the non-object-oriented data store works, and must still manually code his or her software objects so that they can persist to such data stores using an external bridging mechanism. For these and other reasons, therefore, there is a need for the present invention.

Summary of Invention

[0009] The invention relates to object persistence to a relational database, within a run-time environment that has reflection and attribute properties. The reflection property enables discovery of persistence attributes embedded within domain object classes of applications. The invention can be implemented within a software module or library that contains a persistent base class that includes persistence functionality relative to a relational database. A software object class is derived from the persistence base class. The software object class inherits the persistence functionality from the persistence base class. A developer can add appropriate persistence attributes to the software object class source code. The software object class is then compiled to an executable form that has the persistence functionality built-in. The executable software object is thus able to persist to the relational database using this methodology.

[0010] Embodiments of the invention provide for advantages over the prior art. An external bridging mechanism between the object-oriented software objects and the non-object-oriented data store, such as a relational database, is not needed. A developer can have his or her objects persist themselves to the data store, without having to know how to translate object states and relationships into commands and/or languages understandable by the data store. Development of software objects that persist themselves to data stores is thus made simpler and more efficient. Still other advantages, aspects, and embodiments of the invention will become apparent by reading the detailed description that follows, and by referencing the accompanying drawings.

Brief Description of Drawings

[0011] The drawings referenced herein form a part of the specification. Features shown in the drawings are meant as illustrative of only some embodiments of the invention, and not of all embodiments of the invention, unless otherwise explicitly indicated, and implications to the contrary are otherwise not to be made.

[0012] FIG. 1 is a diagram of a run-time environment according to a preferred embodiment of the invention.

[0013] FIG. 2 is a flowchart of a method that is more detailed than but consistent with the method of FIG. 1, according to an embodiment of the invention.

[0014] FIG. 3 is a diagram of a detailed example, in conjunction with which the method of FIG. 2 is illustratively described but to which the method of FIG. 2 is not limited, according to an embodiment of the invention.

[0015] FIG. 4 is a diagram showing an example of how the objects of the object-oriented model resulting from performing the method of FIG. 2, an example of the performance of which is shown in FIG. 3, maps and persists to tables of a relational database, according to an embodiment of the invention.

[0016] FIG. 5 is a diagram showing an example of how the objects of the object-oriented model of FIG. 4 are hierarchically related to the persistent base software object class to obtain persistent functionality, according to an embodiment of the invention.

[0017] FIG. 6 is a diagram showing an example of software code that can utilize domain object classes having persistence functionality built-in, according to an embodiment of the invention.

Detailed Description

[0018] In the following detailed description of exemplary embodiments of the invention, reference is made to the accompanying drawings that form a part hereof, and in which is shown by way of illustration how specific embodiments of the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice them. Other embodiments may be utilized, and logical, mechanical, and other changes may be made without departing from the spirit or scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the invention is defined only by the appended claims.

[0019]

FIG. 1 shows a run-time environment 102 according to a preferred embodiment of the invention. The environment 102 preferably supports attributes and reflection, such as the Microsoft .NET environment, available from Microsoft Corp., of Redmond, Wash. Attributes are keyword-like descriptive declarations, called attributes, which

annotate programming elements, such as types, fields, methods, properties, and so on. Attributes may be saved within the meta data of a Microsoft .NET framework file, and can be used by developers to describe software code to the run-time environment, or to affect application behavior at run-time. The reflection feature allows the environment 102 to discover, or retrieve, attribute or type information.

[0020] The environment has a persistence library 104, and two example application programs 106 and 108. The persistence library 104 is preferably a dynamically linked library (DLL) file. The persistence library 104 has core classes including a persistence base class, a persistence object collection class, and persistence attribute classes. The persistence base class may be referred to as DBObj, whereas the persistence object collection class may be referred to as DBObjCollection. The persistence attribute classes can include PersistenceClassAttribute, PersistenceFieldAttribute, PersistenceObjAttribute, and PersistenceObjCollectionAttribute.

[0021] The persistence base class includes the structure and the methods to be inherited by derived classes to perform persistence functions on instances of the derived classes. The persistence object collection class i.e., the persistence object class is a specialized class, to hold a group of instances of a persistence base derived type. The persistence attribute classes are various attribute types that developers can use to add persistence information to their own domain object classes. The domain object classes typically represent business entities that need persistent storage, and client programs that contain the application logic to manipulate instances of the domain object classes. For example, such business entities may include order forms, customers, order items, and so on.

[0022] The application programs 106 and 108 thus include client programs 114 and 118, respectively, and domain object classes 112 and 116, respectively. The domain object classes 112 and 116 inherit persistence functionality from the persistence library 104, and instances thereof are persisted using this functionality to the relational database management system (RDBMS) 110. The client programs 114 and 118 therefore include the application logic of the application programs 106 and 108 to manipulate instances of the domain object classes 112 and 116, which are retrieved from and saved to the RDBMS 110.

[0023] FIG. 2 shows a method 400 according to an embodiment of the invention, which can be used in conjunction with the environment 102 of FIG. 1. At least some parts of the method 400 can be implemented as a computer program stored on a computer-readable medium. The computer-readable medium may be a volatile or a nonvolatile storage medium, as well as a removable or a fixed medium. The medium can be a magnetic medium, such as a floppy disk or a hard disk drive, an optical medium, such as a CD-ROM, a semiconductor memory medium, or another type of medium. The program may include one or more software components, modules, subroutines, objects, and so on.

[0024] The method 400 is described in particular relation to the example of FIG. 3, for illustrative clarity. However, the method 400 is not limited to the example of FIG. 3. Similarly, other embodiments of the invention are not limited to the example of FIG. 3. FIG. 3 is specifically an example of a domain object class that includes built-in persistence functionality on account of being subclassed, or derived, from a persistence base class, and that can be compiled to yield one or more executable software objects.

[0025] The method 400 of FIG. 2 first references the persistence library (402), which is indicated in FIG. 3 with the reference number 502. The persistence library is the library 104 of FIG. 1. Thus, domain object classes are subclassed from the persistent base class to inherit the persistence functionality (404), as indicated in FIG. 3 with the reference number 504. Next, the persistent class attribute is added to specify the database and the table to which this type maps (406), as indicated in FIG. 3 by the reference number 506. In the example of FIG. 3, this is the persistence information for the class Corderform, specifying the corresponding table in the database that can be referenced by ConnectionString.

[0026] A persistent field attribute is added to object fields that require persistent (408), as indicated in FIG. 3 by the reference number 508. This is the persistence information for field-to-column mapping, specifying the table column to which to map. This information also specifies whether the field is a key field, whether the value is generated by the relational database when initially saved, and whether this field is required. A persistent object attribute is added to objects that are also subclassed

from the persistent base class (410), as indicated in FIG. 3 by the reference number 510. This is the persistence information for field-to-object mapping. The information specifies the column in the current object's table used to fetch the key value used to retrieve a related object, Ccustomer. The information also specifies whether the related object will be deleted when the current object invokes its delete function.

[0027] Next, a persistent collection attribute is added to objects of a particular type (412), which is indicated in FIG. 3 by the reference number 512. In the example of FIG. 3, objects that are of the type DBObjCollection have this attribute added thereto, where this type is a collection of objects that are of the same type that are derived from the persistent base class. Instantiation is needed in the constructor method of the containing class. The persistent collection attribute is thus persistence information for a collection of related objects of the same type, and specifies whether, in the example of FIG. 3, the orderitems related to the orderform will be deleted when the orderform is deleted. The class constructor in the example of FIG. 3 specifies the relationship "orderitems whose orderID equals my orderID, as well as the sort order." The method 400 finally creates the database structure that serves as the persistence storage of domain objects (416). FIG. 4 shows the resulting object-oriented model 602 for the example of FIG. 3, and a corresponding relational database 604 to which the objects of the model can be persisted, according to an embodiment of the invention. The database 604 has a structure that is constructed in 414 of the method 400 of FIG. 2, for instance. The object model 602 includes the objects 606, 612, and 620, whereas the relational database 604 includes tables 610, 616, and 624.

[0028] The customer object 606 is mapped and persisted to the customer table 610, as indicated by the bi-directional arrow 608. Similarly, the orderform object 612 is mapped and persisted to the orderform table 616, as indicated by the bi-directional arrow 614, and the orderitem object 620 is mapped and persisted to the orderitem table 624, as indicated by the bi-directional arrow 622. The relationship between the orderform domain object 612 and the customer domain object 606 is one to one, as indicated by the arrow 618, whereas the relationship between the orderform domain object 612 and the orderitem domain object 620 is one to many, as indicated by the arrow 620. That is, for each order form there is only one customer, whereas there can be more than one order item within each order form.

[0029] FIG. 5 shows an example of the hierarchy of the objects 606, 612, and 620 of FIG. 4 relative to the persistent base software object class 702, according to an embodiment of the invention. That is, FIG. 5 shows the object hierarchy of the example of FIG. 3, with the objects of FIG. 4. Each of the objects 606, 612, and 620 have object types derived from the persistent base class 702, DBObj. Thus, objects of this type can analyze themselves for persistence information, and generate corresponding calls to the relational database 604 of FIG. 4 when persistence functions, such as load, save, and delete, are invoked.

[0030] FIG. 6 shows an example of software code 800 that uses the domain objects that have been described, according to an embodiment of the invention. As can be appreciated by those of ordinary skill within the art, the software code 800 is for illustrative purposes only, and does not represent a limitation on the invention. That is, the invention itself is not limited to the software code 800. The software code 800 includes the portions 802, 804, and 806, among other software code portions.

[0031] The software code portion 802 shows how the method Save() of the class newCust is inherited from the persistence base class. Likewise, the software code portion 804 shows how the method Load() of the class oldCust is inherited from the persistence base class. Specifically, an object that has been persisted to the RDBMS is loaded. Finally, the software code portion 806 shows how the method Delete() of the class oldCust is inherited from the persistence base class.

[0032] It is noted that, although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement is calculated to achieve the same purpose may be substituted for the specific embodiments shown. This application is intended to cover any adaptations or variations of the present invention. Therefore, it is manifestly intended that this invention be limited only by the claims and equivalents thereof.